# Analisis Sentimen menggunakan Neural Network dan LSTM

Hendri Hermawan
Dwinda Ayu Prihantini

# Latar Belakang

Dalam beberapa tahun terakhir, pertumbuhan pesat internet dan media sosial telah menghasilkan jumlah data teks yang sangat besar. Data ini mencakup berbagai jenis informasi, termasuk opini, ulasan, dan komentar yang dapat dianalisis untuk memahami sentimen masyarakat.

Analisis sentimen adalah proses mengidentifikasi dan mengkategorikan opini atau emosi pengguna terhadap suatu topik menjadi positif, negatif, atau netral. Analisis sentimen bisa sangat berguna bagi suatu perusahaan atau organisasi. Perusahaan atau organisasi dapat memahami bagaimana pelanggan mereka merespon produk, sehingga mereka bisa menentukan arah bisnis atau kebijakan apa yang mesti dijalankan.

# Rumusan Masalah

1. Berapa tingkat akurasi rata-rata menggunakan analisis MLP Classification?

2. Berapa tingkat akurasi rata-rata menggunakan analisis LSTM?

3. Bagaimana kesimpulan perbandingan analisis sentiment menggunakan MLP Classification dengan LSTM?

# Metode Penelitian

## MLP Classifier

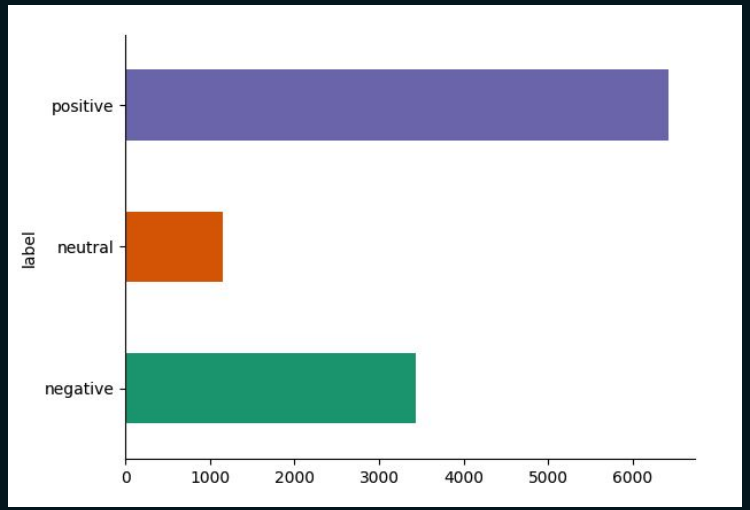
- MLP adalah jenis jaringan neural yang terdiri dari beberapa lapisan neuron. MLP terdiri dari beberapa lapisan dan setiap lapisan terhubung sepenuhnya ke lapisan berikutnya.
- Tools untuk model MLP menggunakan sklearn

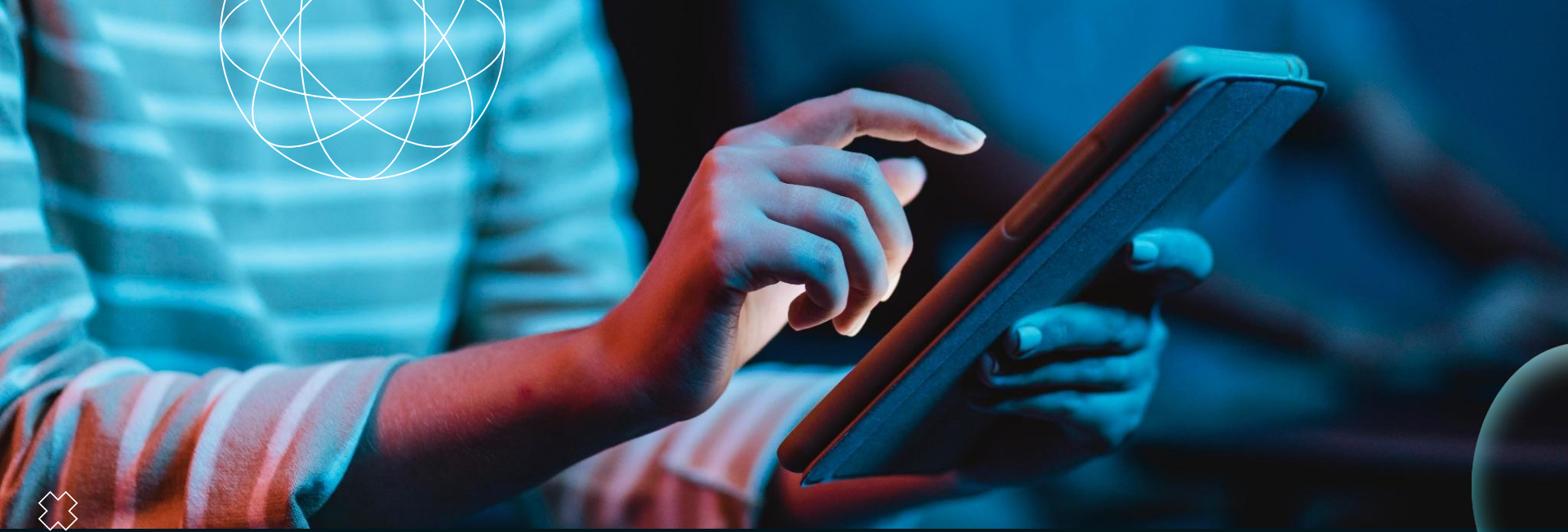
## LSTM

- LSTM adalah jenis Recurrent Neural Network (RNN) yang dirancang untuk mengatasi masalah vanishing gradient pada RNN tradisional. LSTM memiliki kemampuan untuk mengingat informasi dalam jangka waktu yang lebih lama, yang sangat berguna dalam analisis teks yang membutuhkan pemahaman konteks dari urutan kata.
- Tools untuk model LSTM menggunakan tensorflow keras

# Data Analysis

Data sumber berasal dari Binar Academy yang ada di tautan berikut
https://drive.google.com/drive/folders/1qm50pKAnzGGw9gtk--lmwR758aZOVish
Dengan bentuk data 11000 rows × 2 columns





# 01 MLP Classification

# MLP: Preprocessing Data

## 1 Data Cleansing

Data cleansing yang dilakukan meliputi: casting ke lowercase, hilangkan karakter spesial dan simbol

## 2 Replace Kata Alay

Mengganti kata yang mengandung alay menggunakan kamus_alay

## 3 Convert Label Menjadi Numeric

Mengganti text pada kolom ‘label’ menjadi bentuk angka.
Neutral -> 0
Positive -> 1
Negative -> 2

# MLP: Feature Extraction TFIDF

Term Frequency (TF) mengukur seberapa sering sebuah kata muncul dalam sebuah dokumen. Inverse Document Frequency (IDF) mengukur seberapa penting sebuah kata dalam keseluruhan korpus.

TFIDF adalah penggabungan dari TF dan IDF sehingga memberikan penilaian yang baik. Pada penelitian ini, kami sudah mencoba TFIDF dan Bag of Words, dan hasilnya TFIDF memiliki akurasi yang lebih baik.
Feature Extraction TFIDF menggunakan modul yang ada pada sklearn yaitu TfidfVectorizer.

# MLP: Modeling

Data kemudian kami pisah menggunakan train_test_split dari sklearn, 80% training dan 20% untuk validasi
Model training menggunakan MLPClassifier dari sklearn dengan konfigurasi sebagai berikut:

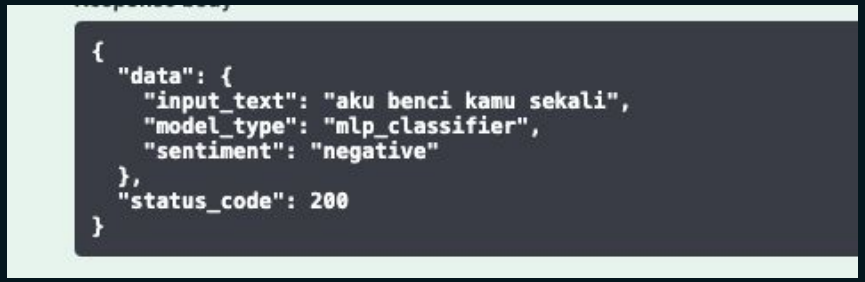
```
mlp = MLPClassifier(
      hidden_layer_sizes=(40),
      max_iter = 100,
      learning_rate_init= 0.1,
      random_state = 123,
      early_stopping=True,
      warm_start = True,
      verbose=True
  )
mlp.fit(X_tr, y_tr)
y_pred = mlp.predict(X_tt)
score = accuracy_score(y_tt, y_pred)
score
```

# MLP: Training & Evaluation

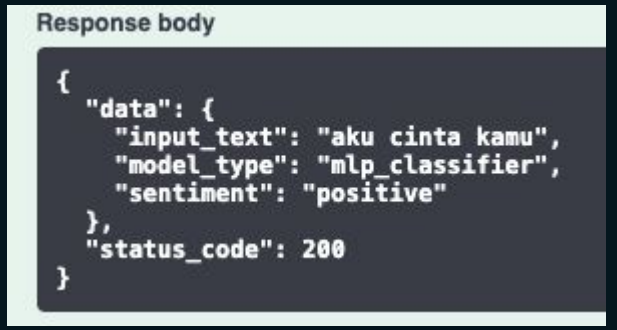
Hasil evaluasi dari 5 kali training menggunakan KFold, didapatkan bahwa rata-rata akurasi sebesar 0.86

```
====================================
Training ke-5
              precision    recall  f1-score   support

           0       0.78      0.73      0.75       245
           1       0.91      0.90      0.91      1285
           2       0.79      0.83      0.81       670

    accuracy                           0.86      2200
   macro avg       0.83      0.82      0.82      2200
weighted avg       0.86      0.86      0.86      2200

====================================


Rata-rata Accuracy:  0.8623636363636364
```

# MLP: Test

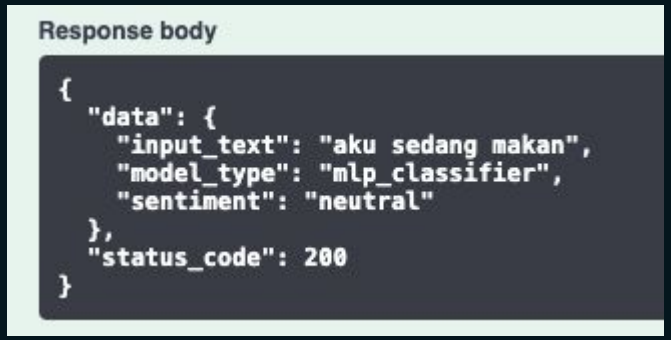


```
{
  "data": {
    "input_text": "aku benci kamu sekali",
    "model_type": "mlp_classifier",
    "sentiment": "negative"
  },
  "status_code": 200
}
```



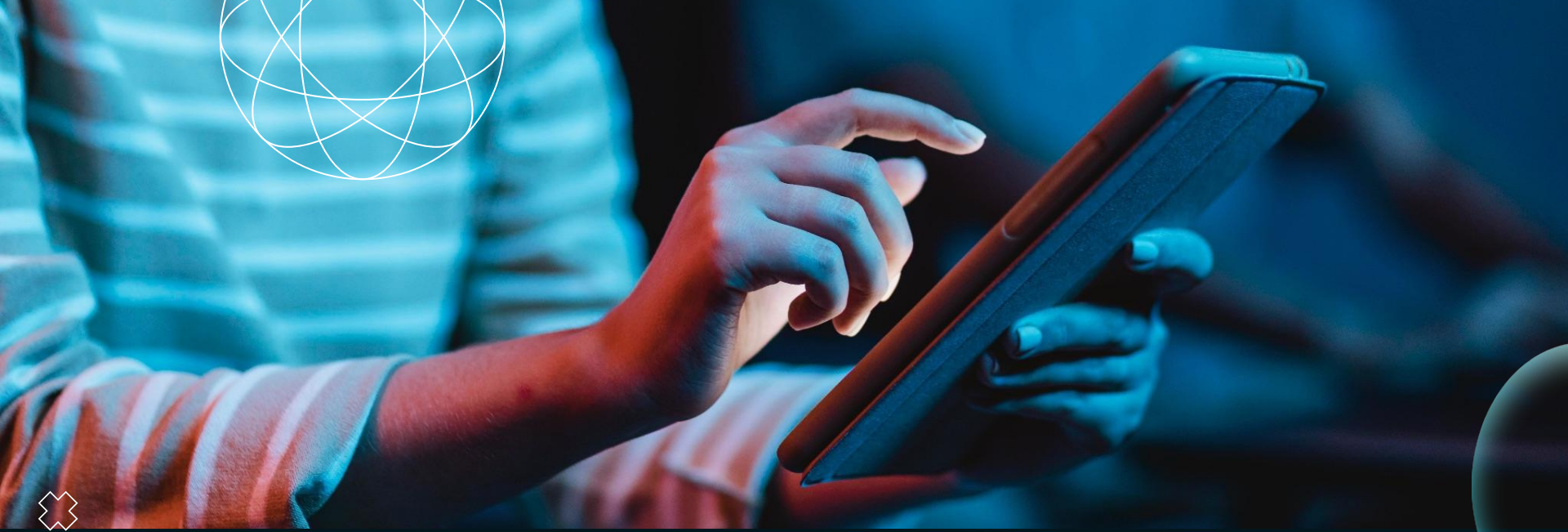
Response body

```
{
  "data": {
    "input_text": "aku cinta kamu",
    "model_type": "mlp_classifier",
    "sentiment": "positive"
  },
  "status_code": 200
}
```



Response body

```
{
  "data": {
    "input_text": "aku sedang makan",
    "model_type": "mlp_classifier",
    "sentiment": "neutral"
  },
  "status_code": 200
}
```



# 02 LSTM

# LSTM: Preprocessing Data

## 1 Data Cleansing

Data cleansing yang dilakukan meliputi: casting ke lowercase, hilangkan karakter spesial dan simbol

## 2 Replace Kata Alay

Mengganti kata yang mengandung alay menggunakan kamus_alay

## 3 Convert Label Menjadi Numeric

Mengganti text pada kolom ‘label’ menjadi bentuk angka.
Neutral -> 0
Positive -> 1
Negative -> 2

# LSTM: Feature Extraction

Menggunakan “Tokenizer” dan “pad_sequences” untuk melakukan feature extraction. Tokenizer digunakan untuk memecah teks menjadi bagian yang lebih kecil atau token. Pad_sequences digunakan untuk memastikan input sequences memiliki panjang yang sama dengan padding nilai tertentu (nol atau tanda khusus).

Dua modul tersebut memungkinkan untuk mengubah list dari sequence (urutan) menjadi array.

# LSTM: Modelling

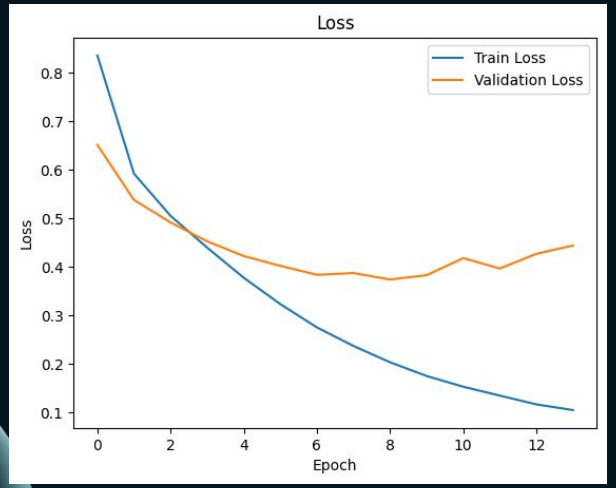
Data kemudian kami pisah menggunakan train_test_split dari sklearn, 75% training dan 25% untuk validasi. Selanjutnya train model dengan tensorflow keras.

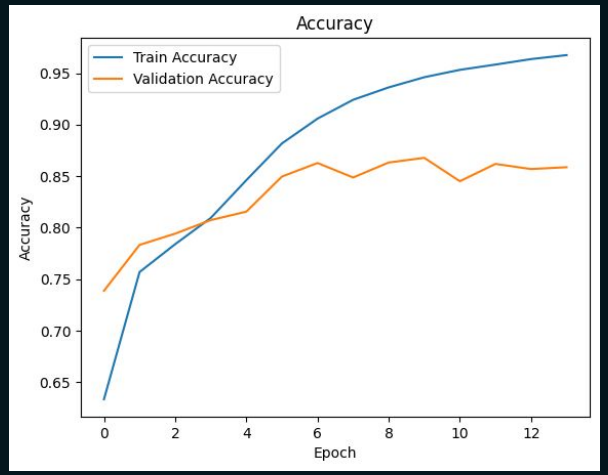
```
[42] embed_dim = 100
     units = 64

     model_lstm = Sequential()
     model_lstm.add(Embedding(input_dim=max_features, output_dim=embed_dim, input_length=X_train.shape[1]))
     model_lstm.add(LSTM(units, dropout=0.2))
     model_lstm.add(Dense(3, activation='softmax'))

     loss_fn = tf.keras.losses.CategoricalCrossentropy(from_logits=False)

     adam = tf.keras.optimizers.Adam(learning_rate=0.0001)
     model_lstm.compile(optimizer=adam,
                  loss=loss_fn,
                  metrics=['accuracy'])

     es = tf.keras.callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=2, patience=5)
     model_lstm.summary()
```

# LSTM:Training Model

Training model didapatkan accuracy score sebesar 0.85

```
69/69 [==============================] - 1s 4ms/step
Testing selesai
              precision    recall  f1-score   support

           0       0.79      0.79      0.79       245
           1       0.90      0.89      0.90      1285
           2       0.78      0.80      0.79       670

    accuracy                           0.85      2200
   macro avg       0.82      0.83      0.83      2200
weighted avg       0.85      0.85      0.85      2200
```




Pada saat modelling, didapatkan grafik “Loss” dan “Accuracy” overfitting.

Hal tersebut kemungkinan dikarenakan data terlalu kompleks.

# LSTM: Test

```
[ ]  input_text = """
     fuck.
     """

     def cleansing(sent):
       string = sent.lower()
       string = re.sub(r'[^a-zA-Z0-9]', ' ', string)
       return string
```

```
[ ]  sentiment = {0: 'neutral', 1: 'positive', 2: 'negative'}

     text = [cleansing(input_text)]
     predicted = tokenizer.texts_to_sequences(text)
     guess = pad_sequences(predicted, maxlen=max_len) #X.shape[1]

     model = load_model('my_lstm_model.keras')
     prediction = model.predict(guess)
```

```
[ ]  polarity = np.argmax(prediction[0])

     sentiment_label = sentiment[polarity]

     print("Text: ",text[0])
     print("Sentiment: ", sentiment_label)
```

```
Text:  fuck
Sentiment:  negative
```

```
Text:  rasa syukur  cukup
Sentiment:  positive
```

# LSTM: Test

Response body


```
{
  "data": {
    "input_text": "deklarasi pemilihan kepala daerah berjalan aman",
    "model_type": "lstm",
    "sentiment": "neutral"
  },
  "status_code": 200
}
```


Response body


```
{
  "data": {
    "input_text": "saya benci sekali",
    "model_type": "lstm",
    "sentiment": "negative"
  },
  "status_code": 200
}
```


Response body


```
{
  "data": {
    "input_text": "saya sangat suka minum kopi",
    "model_type": "lstm",
    "sentiment": "positive"
  },
  "status_code": 200
}
```

# KESIMPULAN

# Kesimpulan

1. Analisa MLP Classification menggunakan tools dari sklearn memiliki rata-rata tingkat akurasi sebesar 0.86 atau 86%

2. Analisa LSTM menggunakan tools dari tensorflow memiliki rata-rata tingkat akurasi sebesar 0.85 atau 85%

3. MLP Classification membutuhkan load time yang lebih cepat dibandingkan dengan LSTM, dan ukuran yang lebih ringan dari LSTM. Namun untuk mendapatkan modeling yang kompleks dengan akurasi yang lebih tinggi, LSTM bisa menjadi pilihan yang tepat.

# Thanks!

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**